Videojuegos

Curso de Diseño y Programación

Nº 15 5,99 euros

Editamos las ***pistas MIDI***

Manejo de ***dispositivos de entrada***

Grabación del ***movimiento de la cámara***









Curso práctico de Videojuegos

AUTOR DE LA OBRA
Marcos Medina
DIRECCIÓN EDITORIAL
Eduardo Toribio
etoribio@iberprensa.com
COORDINACIÓN EDITORIAL
Eva-Margarita García
eva@iberprensa.com
DISEÑO Y MAQUETACIÓN
Antonio Gª Tomé
PRODUCCIÓN
Marisa Cogorro
SUSCRIPCIONES
Tel: 91 628 02 03
Fax: 91 628 09 35
suscripciones@iberprensa.com
FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: Gráficas Don Bosco
DUPLICACIÓN CD.ROM: M.P.O.
DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Iberprensa
www.iberprensa.com

CONSEJERO
Carlos Peropadre

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, 7 (Pol. Ind. "El Nogal")
28110 Algete (Madrid)
Tel.: 91 628 02 03
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-35934-2002
ISBN: Coleccionable: 84 932417 2 5
Tomo 2: 84 932417 4 1
Obra Completa: 84 932417 5 X

NOTA IMPORTANTE:
Algunos programas incluidos en los CD de "Programación y Diseño de Videojuegos" son versiones completas, pero en otros casos se trata de versiones demo o trial, versiones de evaluación que Iberprensa quiere ofrecer a nuestros lectores. No se trata en ningún caso de las versiones comerciales de los programas, y las hemos incluido para dar al lector la oportunidad de conocer y probar esos programas y que así pueda decidir posteriormente si desea o no adquirir las versiones comerciales de cada uno.

Aprende divirtiéndote

Bienvenidos de nuevo a **Programación y Diseño de Videojuegos**, la primera obra coleccionable cuyo objetivo es formar al alumno en las principales técnicas relacionadas en el desarrollo completo de un videojuego.

A lo largo de la obra el lector está aprendiendo programación a nivel general y a nivel específico con ciertas herramientas y lenguajes, aprendiendo a trabajar con aplicaciones de retoque de imagen y también de diseño 3D y animación. Estamos descubriendo las aplicaciones profesionales más importantes de audio y conociendo la historia de lo que se denomina "la industria del videojuego", los últimos 20 años, los juegos que marcaron un avance, sus creadores y en general la evolución del videojuego.

Pero además, esta obra tiene un segundo objetivo, desarrollar y potenciar la creatividad del lector, nosotros a lo largo de las diferentes entregas pondremos las bases y tú pondrás tu ingenio, tu creatividad y tu capacidad de mejorar.

Nos encontramos a mitad de camino del viaje de 20 semanas que os proponemos, viaje articulado en 400 páginas y 20 CD-ROMs cuya finalidad es proporcionar las bases mínimas para después cada uno continuar su camino.

Recuerda que para alcanzar el éxito necesitas cumplir tres condiciones: que te gusten los juegos, poseer cierta dosis de creatividad y finalmente capacidad de estudio.

Una la cumples seguro.

sumario

15

PARA ENCUADERNAR LA OBRA:

- *Tapas del volumen 1 disponibles en Iberprensa. Pedidos por teléfono: 916280203*
- ***Tapas del volumen 2 ya a la venta en quioscos.***
- ***Los suscriptores recibirán las tapas en su domicilio sin cargo alguno como obsequio de Iberprensa.***

SERVICIO TÉCNICO:
Para consultas, dudas técnicas y reclamaciones Iberprensa ofrece la siguiente dirección de correo electrónico: games@iberprensa.com

PETICIÓN DE NÚMEROS ATRASADOS:
El envío de números sueltos o atrasados se realizará contra reembolso del precio de venta al público más el coste de los gastos de envío. Pueden ser solicitados en el teléfono de atención al cliente 91 628 02 03

Dando vida al terreno de **juego**

Después de trabajar con el sistema de partículas del juego, pasamos a desarrollar las funciones necesarias para situar todos los elementos del decorado sobre el terreno de combate. Explicaremos el funcionamiento de los dos modos de zona de combate: el aleatorio y el editado.

PETICIÓN DESDE EL MENÚ PRINCIPAL

La elección del tipo de terreno del juego viene determinada por la elección del jugador en el menú principal. Nos situamos en la función "Panel_opciones()" del módulo "Menu.bb". En la opción 0 de la estructura "Select...Case", correspondiente a un nuevo juego, preguntamos al usuario qué tipo de juego desea de entre los dos disponibles:

```
Case 0:  ; Opción nuevo juego
  ...
  Repeat
    ...
    Tipo_juego=Input$(" ")
    If Tipo_juego=0 Then Return
  Until Tipo_juego=1 Or Tipo_juego=2
  ...
```

Con la información almacenada en la variable global "Tipo_juego" ya sabemos qué procedimientos debemos activar para crear el decorado del juego. Cada uno de estos procedimientos lo encontramos en el módulo "funcpantaudio.bb" en la función "Crear_Decorado()".

La función "Crear_Decorado()" es la encargada de llamar a cada uno de los procedimientos necesarios para situar todos los elementos del juego sobre el terreno (construcciones, árboles, rocas, plantas y animales). Se basa en dos estructuras "Select..Case" anidadas, las cuales dirigen la función a los diferentes procesos. La estructura principal selecciona el tipo de juego a través de la variable global "Tipo_juego". Al comienzo de la función "sembramos una semilla" para la generación de números aleatorios diferentes cada vez que se llame a la función:

```
Function Crear_Decorado()
  SeedRnd MilliSecs()
  Select Tipo_juego
    Case 1: ; Zona de juego
    generada por ordenador
  ...
```



NOTA

De cómo el programa carga y prepara el terreno según el tipo de juego elegido se estudió en el número 9 de esta sección "Visualizando el terreno de juego".

SITUANDO EL DECORADO ALEATORIAMENTE

Si "Tipo_juego" vale 1, el jugador ha elegido generar el decorado aleatoriamente. En el diseño del juego se ha contemplado también distinguir entre los dos tipos de terrenos para generar zonas de combate diferentes. Así que entramos en otra estructura "Select..Case" con las dos opciones de terreno disponibles según el valor contenido en la variable global "Tipo_terreno" (Ver Código 1).

Código 1. Estructura "Select...Case" con las dos opciones de terreno

```
Select Tipo_juego
  Case 1: ; Zona de juego generada por ordenador
    Select Tipo_terreno
      Case 1:
        colocar_almacen1(mesh_almacen1,12000,3000,14000,7500,2*Densidad_decorado,600)
        colocar_almacen2(mesh_almacen2,10000,5000,12000,7500,2*Densidad_decorado,600)
...
  Case 2:
        colocar_arbol1(mesh_arbol1,12000,3000,14000,7500,2*Densidad_decorado,200)
        colocar_arbol2(mesh_arbol2,10000,5000,12000,7500,2*Densidad_decorado,200)
...
End Select
```

Para ejecutar el proceso de colocación es preciso seguir unas pautas si queremos ubicar cada elemento correctamente en su sitio. Por ello, vamos a utilizar una función para cada uno de ellos que hará el trabajo según unos valores predefinidos (Ver Fig.1).

(■) COLOCANDO EL DECORADO

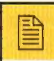
Vamos a utilizar una estructura de datos para cada tipo de objeto para definir su posición. Así que las creamos en el módulo de definiciones "definiciones.bb". Por ejemplo, para los puentes tenemos:

```
Type tipo_puente
  Field x_decorado#,z_decorado#
  Field entidad_decorado
End Type
```

Podíamos haber utilizado matrices (arrays), pero siempre es mejor usar estructuras porque es más rápido, elegante y, a priori, fácil de utilizar para manejar varios datos por objeto. En la estructura se observa que solo contemplamos las coordenadas X y Z, eso es así porque la coordenada Y (altura) la calculamos automáticamente en el momento de colocar el objeto y depende de la altura del terreno, la cual obtenemos con la función "TerrainY". En el módulo "Decorado.bb" instalamos las funciones que sitúan todo el decorado sobre el terreno de juego. Añadimos una función para cada tipo de objeto, la cual agrega un nuevo elemento a la estructura y lo posiciona en el terreno. Vamos a tomar como ejemplo la función para colocar los almacenes de tipo 1 "colocar_almacen1()", ya que la estructura de las demás funciones es exactamente igual. Nuestro propósito para evitar tener que colocar uno a uno cada elemento es disponer de un procedimiento dentro de la función que lo haga automáticamente. Evidentemente, es necesario que le proporcionemos todas las variables necesarias a través de los parámetros de la función. Así que vamos a necesitar los siguientes (Ver Fig.2):

```
Function colocar_almacen1
(entidad,x1_zona#,z1_zona#,
x2_zona#,z2_zona#,cantidad,
dispersion%)
```

■ **Entidad:** A través de este parámetro le decimos a la función qué modelo queremos colocar. En realidad, no es necesario, ya que tenemos una función por cada uno de ellos. Aun así, mantenemos flexibilidad en la función y una vez terminada

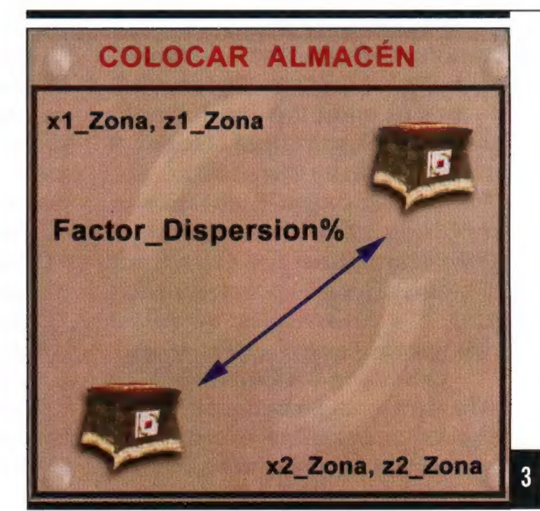


NOTA

Hay que recordar que, sobre el terreno, la coordenada X representa izquierda y derecha y la coordenada Z delante y atrás.

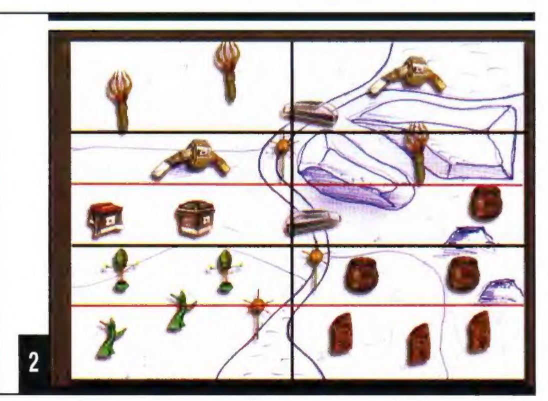
1

Coordenadas de las seis zonas principales de situación del decorado.



3

Esquema del funcionamiento de los parámetros de la función "colocar_decorado()".



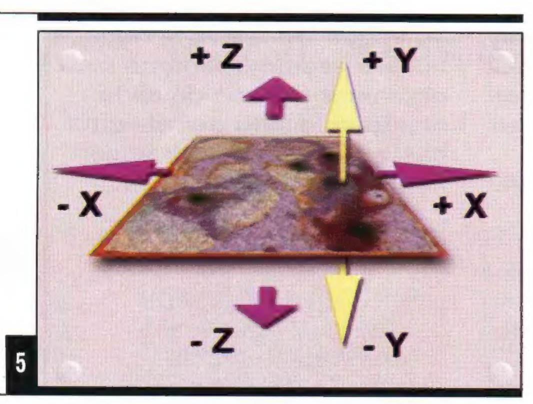
2

En la imagen vemos subdivisiones creadas para colocar cada tipo de decorado.

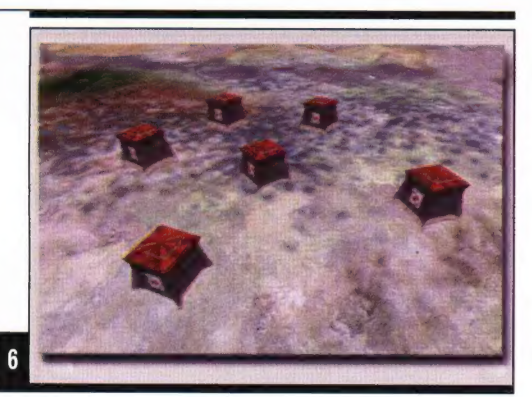


4

Esquema de los datos leídos y utilizados del fichero de zonas.



Tomando como referencia la cámara, en la imagen se muestra la dirección en las distintas coordenadas.



Ejemplo del resultado de aplicar la función para colocar aleatoriamente un modelo.

no necesitaremos volver a entrar para chequearla desde otro módulo.

Para delimitar la zona donde se extenderá la situación del objeto, vamos a definir un rectángulo sobre el terreno. Precisaremos de su posición inicial (definida por la esquina superior izquierda) y su tamaño (esquina inferior derecha). Por lo tanto necesitamos dos coordenadas X y Z para cada punto:

- **X1_zona#:** Coordenada X de la esquina superior izquierda de la zona de colocación.
- **Z1_zona#:** Coordenada Z de la esquina superior izquierda de la zona de colocación.
- **X2_zona#:** Coordenada X de la esquina inferior derecha de la zona de colocación (tamaño).
- **Z2_zona#:** Coordenada Z de la esquina inferior derecha de la zona de colocación.

Luego debemos pasarle a la función la cantidad de objetos que queremos colocar y la separación entre ellos (Ver Fig. 3):

- **Cantidad:** Nº de objetos para situar.
- **Dispersión%:** Separación entre los objetos.

Una vez que tenemos todos los datos necesarios para poder situar los objetos, entramos en la función para realizar la operación. En primer lugar, es necesario crear tantos objetos como hemos indicado en "cantidad". Para ello, entramos en un bucle para ir creando nuevos elementos de la estructura y asignarle a cada uno una copia del *mesh* correspondiente y asignarles la identidad para el sistema de colisiones:

```
For n = 1 To cantidad
  almacen1.tipo_almacen1=
   New tipo_almacen1
  almacen1\entidad_decorado=
   CopyEntity(entidad)
  EntityType almacen1\
   entidad_decorado,
   ENTIDAD_EDIFICIO
Next
```



El editor de niveles muestra la disposición exacta que tendrá el decorado en el juego.

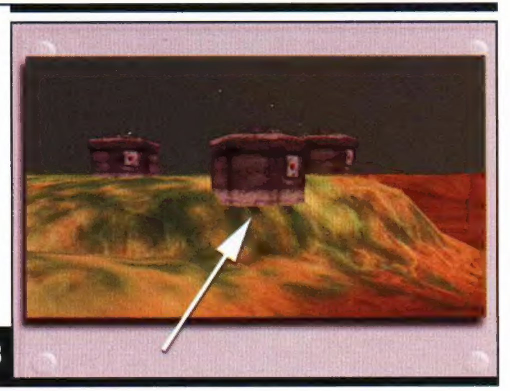
Código 2. Recorremos cada elemento de la estructura

```
For almacen1.tipo_almacen1= Each tipo_almacen1
 m=almacen1\entidad_decorado
 almacen1\x_decorado#=Rnd(x1_zona#,x2_zona#)
 almacen1\z_decorado#=Rnd(z1_zona#,z2_zona#)
 xg#=almacen1\x_decorado#
 zg#=almacen1\z_decorado#
 yg#=TerrainY(terreno1,almacen1\x_decorado#,EntityY(m),almacen1\
     z_decorado#)
 PositionEntity m,xg+factor_dispersion,yg-20,zg+factor_dispersion
 RotateEntity m,0,Rnd(0,360),0
 Factor_dispersion=Rnd(100,dispersion)
Next
```

Una vez que hemos creado todos los almacenes pasamos a situarlos en el terreno. Para ello, recorremos cada uno de los elementos de la estructura con un bucle "For .. = Each" (Ver Código 2).

Almacenamos en "m" la entidad que vamos a colocar para reducir código posteriormente. Seguidamente, asignamos unas coordenadas aleatorias de situación dentro de la zona definida por los valores "x1_zona, z1_zona" y "x2_zona, z2_zona". Luego guardamos en las variables "xg","yg" y "zg" las coordenadas 3D del objeto para poderlo posicionar.

La coordenada Y para la altura se calcula automáticamente dependiendo de la del terreno en la coordenada especificada por "xg" y "zg" aprovechando la instrucción


8

Un ejemplo de error típico derivado de la colocación aleatoria de objetos.

"TerrainY". Para posicionar el almacén vamos a tener en cuenta el factor de dispersión. De esta manera, evitamos que queden demasiados juntos. Para llevar a cabo este sistema, solo tenemos que sumar a las coordenadas X y Z el factor de dispersión que guardamos en la variable "factor_dispersion". Este factor estará comprendido entre 100 y el parámetro "dispersion".

Para terminar, también aplicamos una rotación en el eje Y aleatoria entre 0 y 360 grados con el fin de ganar variedad en el conjunto.

Podemos añadir más parámetros a la función como escalado del modelo, color, efecto del material, etc. También es evidente que el control que tenemos de la situación de los objetos no es muy preciso, ya que se colocan aleatoriamente. Por lo tanto, es muy normal que parte de los objetos del decorado queden en el aire al coincidir con un desnivel. Sin embargo es posible corregir este error controlando la situación de cada vértice del modelo, aunque esto resulta más complicado de implementar. Otra forma de evitar esto es utilizar un sistema más controlado como utilizar zonas editadas previamente como veremos a continuación.

(■) CARGANDO ZONAS EDITADAS

Volviendo a la función "crear_decorado()" del módulo "funcpantaudio.bb", nos situamos en la segunda opción de juego, en donde creamos la zona de combate a partir de los datos contenidos en el fichero que produce el editor de zonas de combate. El sistema es realmente sencillo. En primer lugar, cargamos el fichero con los datos:

```
Case 2: ; Zona de juego editada
 fichero=ReadFile("c:\zone of
 fighters\gfx\datos\
 "+zona_de_combate$)
...
```

Seguidamente, entramos en un bucle "While .. Wend" en el que vamos leyendo el fichero secuencialmente para obtener los datos de cada objeto colocado por el editor hasta que no encontremos ninguno más (Ver Fig. 4):

```
While Not Eof(fichero)
  objeto$=ReadString(fichero)
  null1=ReadByte(fichero)
  null2=ReadFloat(fichero)
  posx#=ReadFloat(fichero)
  posy#=ReadFloat(fichero)
  posz#=ReadFloat(fichero)
  roty#=ReadFloat(fichero)
  null3=ReadByte(fichero)
  ...
```

Entre los datos leídos encontramos las coordenadas de situación (posX, posX, posZ) y el ángulo de rotación (rotY). Además, obtenemos el nombre del objeto que corresponde (objeto$), lo que es necesario para saber qué modelo debemos crear y situar. De tal forma, entramos en una estructura "Select..Case" en la que preguntamos qué objeto crear (Ver Código 3).

El funcionamiento es bien fácil: sólo tenemos que crear y posicionar el objeto dependiendo de los datos obtenidos del fichero.

No olvidemos, después de leer y colocar todos los objetos contenidos en el fichero, cerrarlo con "CloseFile (fichero)".

Como hemos comprobado, implementar un sistema automático de creación de decorados es una tarea sencilla y con buenos resultados, lo que permite que el juego resulte diferente cada vez. Por otro lado, la posibilidad de reproducir exactamente una zona de combate a partir de un editor trasciende en un mayor atractivo para el jugador, el cual podrá participar de lleno en un nivel original y propio.

Código 3. Preguntamos qué objeto crear

```
...
Select objeto$
 Case "almacen1"
  almacen1.tipo_almacen1= New tipo_almacen1
  almacen1\entidad_decorado=CopyEntity(mesh_almacen1)
  EntityType almacen1\entidad_decorado, ENTIDAD_EDIFICIO
  almacen1\x_decorado#=posx#
  almacen1\z_decorado#=posz#
  PositionEntity almacen1\entidad_decorado,posx#,posy#,posz#
  RotateEntity almacen1\entidad_decorado,0,roty#,0
 Case "almacen2"
  almacen2.tipo_almacen2= New tipo_almacen2
  ...
...
```

En el próximo número...

... nos sumergiremos en el sistema de inteligencia artificial del juego.

Fabricando los elementos del juego (VI)

Terminaremos en este número de texturizar los demás elementos del decorado del juego. Pero para llevar a cabo nuestra misión es necesario conocer algunas técnicas para obtener texturas de calidad.

Aparte de dibujar manualmente cualquier textura con un programa de diseño podemos utilizar otras técnicas como el retoque fotográfico o la utilización de aplicaciones 3D. Si, por ejemplo, necesitamos la textura de una pared siempre se puede obtener una fotografía de una de verdad y posteriormente retocarla en Paint Shop Pro o cualquier otra aplicación para que se amolde a nuestras necesidades. Realmente, este sistema es sencillo y no requiere mucho esfuerzo. Además, se consiguen resultados muy buenos ya que la textura parecerá real. Puede ocurrir que unas buenas texturas fotográficas no queden demasiado bien en un modelo con poco polígonos o que el conjunto de estas texturas no esté acorde con el resto de elementos del juego. Aun así, siempre se obtienen resultados estupendos. La utilización de aplicaciones de modelado y renderizado 3D también resulta interesante para la obtención de texturas preiluminadas. En el texturizado del resto de modelos del decorado vamos a utilizar esta técnica utilizando el programa Bryce 5.0 (aunque cualquier versión es válida).

TEXTURIZANDO EL PUENTE

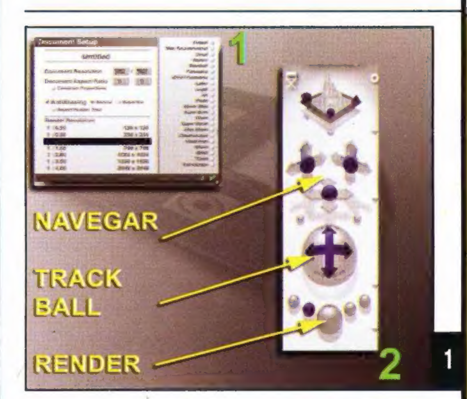
Para obtener la textura que configura el suelo y el techo del puente vamos a utilizar, como hemos dicho, Bryce 5. La idea es texturizar una primitiva como un cubo y aplicarle una textura; a continuación, orientamos el objeto hacia la cámara y grabamos una imagen del renderizado. Esta imagen contendrá la textura que necesitamos. Hay que tener en cuenta que si la textura que creemos con esta técnica va destinada a un tileado, necesitaremos hacerla cíclica con un programa de retoque para que al duplicarla no se note la unión entre los bordes. Sin embargo, en nuestro caso no utilizaremos tileado. Así que nos centraremos en nuestra textura individualmente. Una vez que hayamos obtenido la textura deseada en una imagen pasaremos a Paint Shop Pro para texturizar el puente a través de la plantilla.

OBTENIENDO LA PLANTILLA

Como sabemos, antes de texturizar necesitamos crear el mapeado UV del modelo. Para obtenerlo, utilizaremos el mismo procedimiento que usamos con los almacenes en el programa LithUnwrap. Cargamos el modelo, lo seleccionamos, aplicamos la opción *Tools / UV Mapping / Box*. Guardamos la plantilla con un tamaño de 512 x 512 en *File / Template / Save*.

UTILIZANDO BRYCE

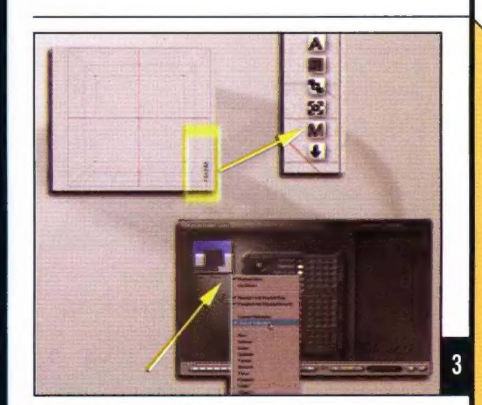
Instalemos Bryce 5 y ejecutemos el programa. No vamos a hacer una explicación exhaustiva de su funcionamiento, solo lo necesario para nuestras necesidades. Para empezar vamos a cam-



Descripción del panel de navegación con las opciones "Navegar", "Track ball" y "Render".



Descripción de las herramientas de crear y editar: podemos especificar las luces, escalar o rotar la imagen, etc.

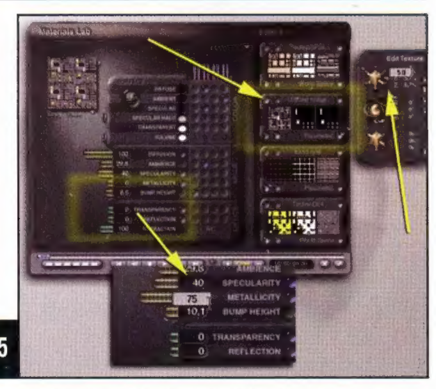


Para ir al laboratorio de materiales pulsamos sobre la letra "M" situada en la parte inferior derecha del objeto.



4

En cada ventana que abrimos en el laboratorio de materiales disponemos de una ventana de previo.



5

En el laboratorio de materiales disponemos de todas las herramientas necesarias para modificar el material elegido.

biar el tamaño del documento. Elegimos la opción *File/Document Setup* y entramos en una ventana de diálogo en la que podemos cambiar la resolución de la pantalla. Deseleccionamos *Constrain Propotions* y en *Document Resolution* escribimos 512 x 512.

En el menú *create* situado en la parte superior, encontramos una serie de elementos que podemos crear. Elegimos el cubo y hacemos clic sobre él. Observamos cómo aparece en el entorno 3D sobre la cuadrícula. Para navegar por el entorno 3D encontramos a la izquierda una serie de iconos que describimos en la figura 1 / 2. Con la *Camara Track Ball* nos situamos en la línea de su cara frontal. (En la figura 2 se muestra una descripción de los iconos del menú *Edit*).

Posteriormente, en el menú *Edit* seleccionamos el icono de desplazamiento para situar el cubo a nuestra altura. Luego, escalamos el cubo con la herramienta de escalado hasta que cubra toda nuestra vista.

Ya casi estamos preparados para entrar en el editor de materiales. Como comentábamos, la idea es texturizar este cubo y el resultado grabarlo en una imagen. Pero antes, vamos a orientar la luz del sol para iluminar la cara del cubo que tenemos enfrente. Entramos en el menú *Sky & Fog*. A la derecha vemos una esfera que indica la posición del sol. Con el ratón situamos el sol en la parte inferior. Podemos ver cómo en el previo se ilumina la imagen. Ahora ya estamos preparados. Para entrar en el editor de materiales pulsamos en el icono con la letra "M" situado en la parte inferior derecha del cubo. Una vez dentro, podemos preparar la ventana de previo situada en la parte superior izquierda. Abrimos la lista de opciones de visualización pulsando en la flechita situada al lado de "paste"y elegimos *Actual Selection*. Entonces se mostrará la escena con nuestro cubo.

Orientamos la cámara del previo hasta situarla enfrente del cubo. Nos acercamos a él hasta que cubra toda la pantalla. A continuación seleccionamos el material que asignaremos al cubo. Para ello, pulsamos en la flechita situada en parte superior derecha del previo (Ver Fig. 3).

Entraremos en otra ventana con todos los tipos de materiales disponibles. Observamos que, de nuevo, disponemos de una ventana de previo. Vamos a elegir un material para la superficie del puente. Elegimos la clase *Miscellaneous* y la textura *Urban Dwelling* (Ver Fig. 4).

En el previo podemos ver nuestro cubo con la textura. Es importante tener el cubo totalmente perpendicular a la vista para así obtener una textura sin perspectiva. Una vez aceptada, volvemos al *Materials Lab*, en donde trabajaremos la apariencia de nuestra textura. Podemos observar a la derecha que el dibujo de la textura está formada por otras cuatro texturas. Cada una de ellas puede ser editada. Por el momento, solo vamos a modificar el tamaño de la segunda. Para ello, pulsamos sobre la pequeña esfera de la esquina superior izquierda para abrir el editor. En los valores X, Y y Z del factor de escala hay que escribir 30 para los tres ejes haciendo clic sobre los números. Para terminar, vamos a añadir un poco de efecto metálico y de altura (bump), asignando el valor 75 en *Metallicity*, 10 en *Bump Height* y aumentemos *Reflection* y *Refraction*. ¡Listo! Pulsamos en "aceptar" (esquina inferior derecha) y volvemos al entorno 3D (Ver Fig. 5).

NOTA

En la esquina superior izquierda del programa se encuentra una pequeña ventana de visualización que muestra un previo del renderizado del entorno actual.

Podemos ver entonces cómo en el previo se muestra una imagen de nuestro cubo ya texturizado. Para renderizar la escena solo tenemos que pulsar sobre la esfera más grande situada debajo de los iconos de navegación. Sin embargo, vamos a aprovechar las cualidades de iluminación que un programa de estas características nos ofrece. Ahora vemos que la textura resulta con un aspecto demasiado plano. Así que cambiemos el color y la posición del sol para obtener algún tipo de sombreado. Para cambiar el color del ambiente, nos situamos en *Sky & Fog* y pulsamos sobre el cuadradito situado debajo de la esfera de posición del sol. Elegimos un color celeste y luego cambiamos la posición del sol para que la luz incida lateralmente (Ver Fig. 6).

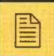
Una vez comprobada la textura pasamos a grabarla en disco. Para realizar este proceso debemos elegir la opción *Render to Disk...* del menú *File* (la barra del menú principal aparece cuando subimos el puntero del ratón hacia la parte superior de la pantalla). El programa nos pedirá la resolución de salida (queda como está) y a continuación el nombre del archivo. Elegimos el formato *.bmp* y guardamos.

Aprovechando la escena vamos a obtener la textura que nos servirá para los soportes curvos del techo del puente, la cual tiene aspecto de piedra. Pulsamos "ESC" y entramos de nuevo en el laboratorio de materiales. Lo único que tenemos que hacer es elegir una textura pétrea de la librería. Por ejemplo, "Arizona" de "Planes&Terrains" y escalarla al 30 %.

COMPLETANDO LA TEXTURA EN PAINT SHOP PRO

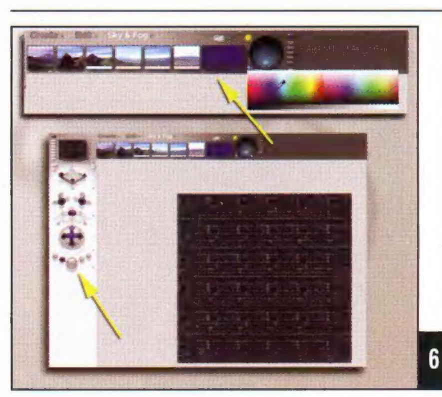
Para completar la textura en Paint Shop Pro emplearemos la misma técnica utilizada para los almacenes. Creamos un documento nuevo de 512 x 512 y cargamos la plantilla con el mapeado UV del puente. Pasemos la imagen de la plantilla a la nueva que hemos creado para tener una capa de referencia. Reduzcamos la opacidad de la capa para volverla semitransparente. Ahora carguemos la primera de las texturas que creamos con Bryce y la pasamos a la imagen principal. Colocamos la nueva capa por debajo de la capa de la plantilla. Seguidamente, con la herramienta de transformación ajustamos su tamaño a la base del puente, el cual corresponde al rectángulo mayor de la segunda vista de la plantilla. Luego, copiamos la capa para cubrir el rectángulo mayor de la quinta vista correspondiente al techo y a la parte superior del suelo. Hagamos lo mismo para el resto de las vistas para cubrir toda la base. No es la forma más correcta ya que, en cada vista, la textura adquiere una apariencia diferente, pero por el momento servirá (Ver Fig. 7).

Recordemos que el techo forma un rectángulo un poco más ancho que la base, así que tenemos que crear dos capas más para adaptarlas a estos rectángulos.

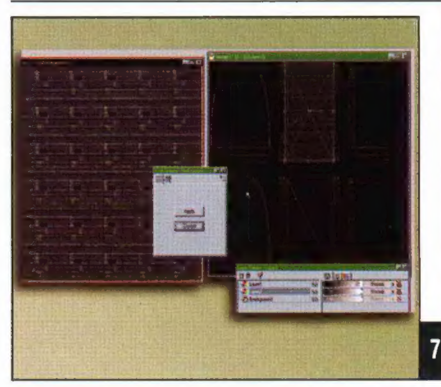


Para los soportes del puente utilizamos la segunda textura que creamos con Bryce. El procedimiento es el mismo. Cargamos la textura y la pasamos a la imagen principal. A continuación, la ajustamos a los soportes de la plantilla. No olvidemos situar la nueva capa por debajo de la capa de la plantilla. El procedimiento es el mismo: duplicar y situar según marque la plantilla (Ver Fig. 8).

Podemos utilizar la herramienta de borrar para ir perfilando aún más las texturas



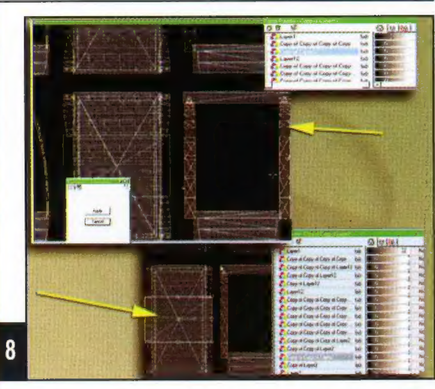
Modificando la luz ambiental y el color podemos modificar el aspecto de la textura.



Una vez obtenida la textura, pasamos a ajustarla en la plantilla.

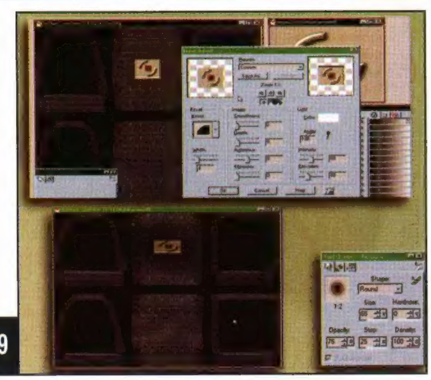
NOTA

Podemos mover y orientar la escena del previo haciendo clic y moviendo el ratón. Para alejar o acercar la cámara dejamos pulsado la tecla "Ctrl" mientras movemos con el ratón. Para rotar la escena con la tecla "Shift" y para desplazar la cámara con la tecla "ESPACIO". Para volver a la vista inicial, pulsar la tecla "Alt".



8

Debemos cubrir toda la plantilla haciendo copias de las capas y ajustando con la herramienta de transformación.



9

Después de colocar la placa del logo debemos retocar la iluminación de la textura para proporcionarle volumen al modelo.



10

La textura del cielo es sencilla de obtener utilizando Bryce.

(de forma rectangular) a las siluetas curvas. Ya solo nos queda colocar el logotipo del juego en el suelo del puente. Aprovechemos el que diseñamos para los almacenes. Lo cargamos y añadimos a la textura principal para aplicarle un efecto de relieve para que parezca que está incrustado en el metal. Ahora debemos retocar la textura con toques de iluminación para diferencia un poco las partes del puente. Pero antes, vamos a unir todas las capas en una sola imagen con la opción *Merge All /Flatten* en el menú que aparece al hacer clic con el botón derecho sobre cualquiera de las capas. Recordemos que hay que eliminar primero la capa de la plantilla, ya que no la necesitamos. Elegimos la herramienta *Retouch* en el modo *Darken RGB* y pintamos en las vistas 1, 3, 4 y 6 las separaciones de las tres partes que forman la base del puente. Esto dará más volumen al modelo (Ver Fig. 9).

Luego hagamos unas líneas longitudinales a lo largo de los soportes del puente. También podemos crear algunas partes más oscuras y claras en el suelo del puente para simular sombras debido a la presencia del techo. Todos los retoques posibles son siempre bienvenidos. Para finalizar, siempre podemos ver el resultado inmediatamente por medio de LithUnwrap.

TEXTURIZANDO LAS ROCAS, ÁRBOLES Y EL CIELO

Como habíamos comentado, las texturas de las rocas y árboles las vamos a realizar utilizando la técnica empleada con Bryce. Para las rocas solo tenemos que elegir la textura pétrea que más nos guste y para los árboles podemos utilizar, si usamos Bryce 5, la nueva librería de texturas para troncos de árboles *Trunks*. Para la realización del cielo vamos a utilizar el renderizado que Bryce tiene de las nubes. Dejemos el entorno sin ningún objeto y apuntamos con la cámara hacia arriba. Nuestra idea es obtener una textura difusa con muchos colores. Así que modifiquemos la estructura de las nubes y tomemos una instantánea. Para modificar el cielo, encontramos todos los parámetros en *Sky & Fog*. Pulsando en la flechita situada a la derecha de *Sky&Fog* entramos en la librería de cielos deBryce. Si usamos la versión 5, una buena opción es el cielo *Middle Of The Sun*. En este apartado podemos apreciar seis ventanitas, cada una con un dibujo de un paisaje. Cada una de ellas representa una opción de modificación del ambiente: sombras, niebla, altura o cantidad de las nubes, etc. La gráfica situada a la izquierda de la esfera de situación del sol representa la frecuencia de aparición de nubes. Haciendo clic y desplazando el ratón podemos modificar su valor. Jugando con todos estos parámetros podemos conseguir el cielo que queramos. Luego solo tenemos que renderizar a disco y ya tenemos nuestra textura. Aunque también podemos texturizar el cubo anterior utilizando la librería de nubes del laboratorio de materiales. Obtendremos texturas más rectas y lineales con este método.

Aprovechando que hemos entrado en el uso de Bryce, vamos a desarrollar el mapa de alturas de nuestro terreno con esta aplicación.



En el próximo número...

... fabricaremos el terreno con Bryce.

Nuestro primer tema musical con Anvil Studio (IV)

En esta entrega terminaremos de aprender la forma de editar las pistas MIDI en la sección *Compose*, modificando los eventos en el panel *Piano Roll*.

Además, entraremos en la creación y edición de pistas de muestras.

EDITANDO EN EL MODO *PIANO ROLL*

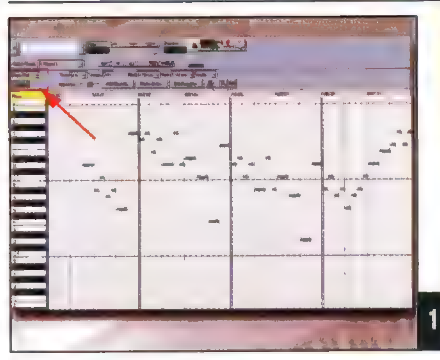
Otra opción de la que disponemos para editar nuestras pistas MIDI es el modo *Piano Roll*. En él se muestra de una forma gráfica cada una de las notas de la pista sobre una cuadricula que representa las divisiones de cada compás. Antes de seguir, carguemos en el programa el tema "FugueGM" habitual. Seleccionamos la primera pista y pasamos a la sección *Compose*. Una vez allí, donde pone *Staff* elegimos *Piano Roll* de la lista. Observamos cómo en la cuadrícula se reparten una serie de rectángulos, los cuales representan cada nota. La longitud de cada rectángulo significa la duración de la nota. Y el tono dependerá de la posición vertical en la cuadrícula. De todas formas, a la izquierda de la ventana podemos ver un teclado que nos ayudará a tener una referencia del tono (Fig. 1).

EDITANDO LAS NOTAS

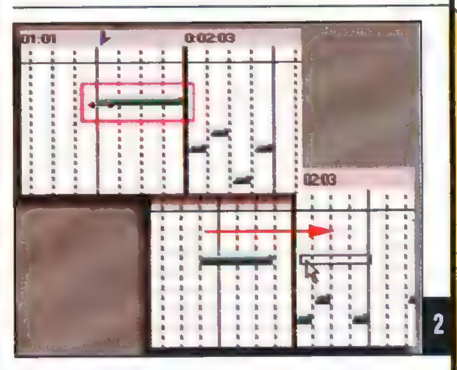
La forma de trabajar con las notas en este modo difiere un poco del modo *Staff*. Para cambiar el tono de una nota (rectángulo), es decir, desplazarla hacia arriba o hacia abajo, nos colocamos sobre ella y la movemos con el ratón (es conveniente elegir una nota de bastante duración para practicar mejor). Si lo que queremos es desplazarla hacia delante o atrás en el compás, debemos situar el puntero del ratón al comienzo de la nota, entonces, éste cambiará de forma. Pasará de la cruz a una flecha de doble sentido. Sin dejar de hacer clic desplazamos el ratón hacia la dirección deseada. Vemos cómo al dejar de hacer clic la nota se adapta automáticamente a la división de la cuadrícula más cercana (cuantización) (Fig. 2).

Para alargar o reducir la duración de la nota, simplemente nos situamos al final del rectángulo con el puntero del ratón; éste, de nuevo, cambiará su aspecto al de una flecha de doble sentido. Sin dejar de hacer clic desplazamos el ratón hacia la derecha o izquierda. En esta ocasión, al dejar de hacer clic, la nota se adaptará a la cuadrícula dependiendo de si tenemos la opción *Snap to Grid* activada o no a través del icono ▣.

Para colocar una nueva nota, es suficiente con hacer clic en cualquier lugar de la cuadrícula. La nota nueva tendrá la duración definida en la casilla "Note". Por defecto adquiere un valor de 1 / 4; es decir, una parte de un 4 por 4. Pulsando con el botón derecho del ratón sobre cualquiera de las notas aparecerá un menú flotante en el que podemos cambiar sus propiedades (duración y volumen) o borrarla (Ver Fig. 3).


1

Modo Piano Roll de la sección Compose.



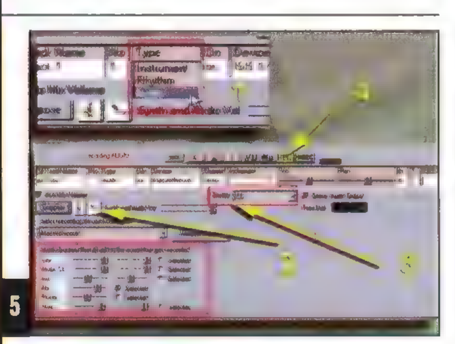
2

Procedimiento para desplazar una nota con el ratón en el tiempo.

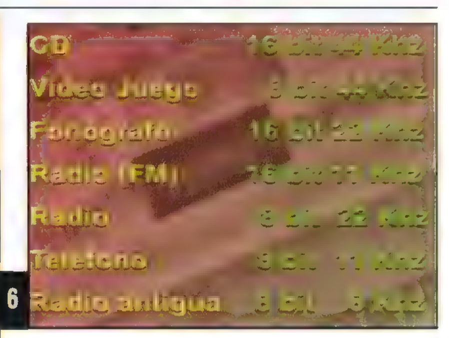


3

La duración de la nueva nota creada dependerá del valor elegido en la casilla "Note".



4 Para seleccionar grupos de notas debemos hacerlo en la línea de tiempo.



5 *Pasos para grabar una pista de audio desde el micrófono.*

CD	16 bit 44 Khz
Video Juego	8 bit 44 Khz
Fonografo	16 bit 22 Khz
Radio (FM)	16 bit 11 Khz
Radio	8 bit 22 Khz
Telefono	8 bit 11 Khz
Radio antigua	8 bit 8 Khz

6 *Esquema de las calidades de audio posibles en Anvil Studio.*



TRUCO

Si queremos ver al mismo tiempo la sección *Compose* en *Piano Roll* y las pistas de *Mixer*, después de reducir la vista, pulsamos en el botón "Mixer" situado bajo *Piano Roll* y luego volvemos a la sección *Compose* pulsando en el botón "Compose".

EDITANDO GRUPOS DE NOTAS

Antes de seguir, vamos a reducir un poco la cuadrícula para poder observar mejor todas las operaciones de edición. Para ello, hacemos clic en el icono de la lupa con el signo menos.

Vamos a copiar un grupo de notas de un lugar a otro. Par seleccionar las notas debemos hacerlo en la línea de tiempo (Ver Fig. 4).

Por ejemplo, seleccionemos un compás entero. Desplazamos el ratón hacia la derecha sin dejar de hacer clic. Después, hacemos "Ctrl" + "C" para copiar.

Nos situamos dos o tres compases después haciendo clic en la línea de tiempo e insertaremos el compás copiado con la opción *Paste Insert* ("Mayús" + "Insert") en el menú *Edit*. Observamos cómo el compás seleccionado al principio se inserta en el lugar elegido. Del mismo modo que seleccionamos un compás entero, podemos trabajar solo con un grupo de notas. Las opciones de copiado funcionan de igual manera que en el modo *Staff*; es decir, insertar, mezclar o solo un evento especial.

OTRAS POSIBILIDADES

Al igual que ocurre con las notas, podemos insertar un silencio situándonos en la línea de tiempo y pulsando en el icono. La duración del silencio dependerá de la opción elegida en la casilla "Note".

Hay una opción muy interesante en Anvil Studio que es la creación y gestión de Loops (bucles) de una manera realmente completa. Pero este tema lo dejaremos para una próxima entrega debido a que haremos un estudio detallado de su funcionamiento y utilización.

PISTAS DE AUDIO

Otra de las características que Anvil Studio admite es la posibilidad de crear y editar pistas de audio. Inicialmente, solo permite una sola pista de audio (mono o estéreo). Si se quiere más, es necesario adquirir el accesorio *Anvil Studio MultiAudio*. De todas formas, con una sola es suficiente para nuestros propósitos.

GRABANDO UNA PISTA DE AUDIO A TRAVÉS DEL MICRÓFONO

Antes de empezar, nos situamos en *Mixer* y creamos una nueva canción en *File / New*. Cambiamos el tipo de pista "instrument" (por defecto) por una de audio. Seleccionamos la tarjeta de sonido pulsando en el icono y configuramos el volumen adecuado del micrófono. Podemos, además, seleccionar si la pista de audio es mono o estéreo en la casilla "Channel". Seguidamente, solo tenemos que pulsar en el icono de grabación "REC". A continuación, el programa pedirá el nombre del nuevo fichero de audio que se va a crear y su ubicación. Una vez proporcionado, el programa empezará a grabar (Fig. 5).

La calidad de la grabación la podemos elegir en la casilla "Quality". En la figura 6 se muestra un esquema con los distintos tipos de calidades soportados por el programa.

GRABANDO UNA PISTA DE AUDIO DESDE EL CD

Si lo que deseamos es importar una pista de CD a la pista de audio, simplemente elegimos la calidad "CD"y pulsamos en el botón "VU". Después de hacer play en el CD pulsamos en "REC".



En el próximo número...

... terminaremos con el sistema de manejo de pistas de audio.

Teclado, ratón y dispositivos de juegos

En todo videojuego o aplicación multimedia es fundamental la comunicación con el usuario.

La interactividad es la esencia del videojuego. Sin ella, el concepto de juego desaparecería. No solo es primordial una comunicación entre ordenador y humano, sino que ésta debe ser rápida, fluida y cómoda. Como todo lenguaje de programación, Blitz3D permite el manejo del teclado, ratón y otros dispositivos de entrada de datos manual como un gamepad o un joystick. En este número trataremos de explicar la utilización de estos elementos.

FUNCIONES DE TECLADO

Hasta que no se perfeccione y estandarice el reconocimiento de voz, el teclado será la única herramienta para introducir caracteres en el ordenador. La manera más conocida de satisfacer las peticiones de un programa es a través de la instrucción "Input":

```
Input (      $)
```

El "prompt" simplemente te indica que el ordenador está listo para recibir datos desde el teclado. Se mantiene a la espera mostrando un cursor, sin embargo, es posible añadir una frase además del cursor. Para utilizar "Input" es necesario asignar la operación directamente a una variable. Dependiendo del tipo de variable, "Input" automáticamente, sabrá qué tipo de datos se va a introducir. Por ejemplo, para obtener una frase:

```
= Input ("
"): Print       $
```

Para escribir un número:

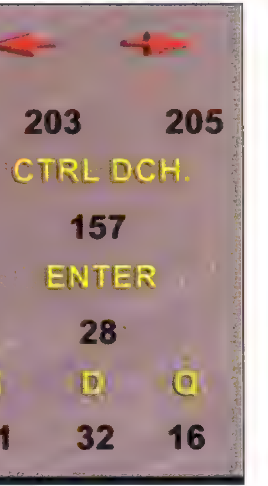
```
= Input ("
")
```



Usos más comunes de las instrucciones "KeyHit()" y "KeyDown()".

"Input" detiene la ejecución del programa a espera de datos y la aceptación con "Return", "GetKey()" no necesita hacerlo. Sin embargo, solo obtiene un carácter a la vez y solo su valor ASCII, el cual será almacenado en una variable.

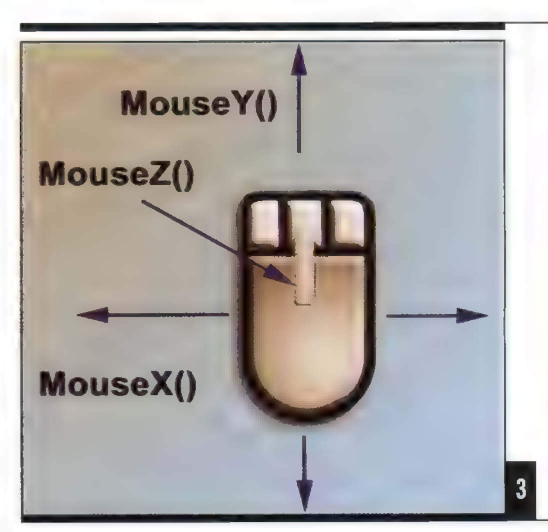
Por lo tanto, esta variable debe ser de tipo numérico: Caracter = GetKey()



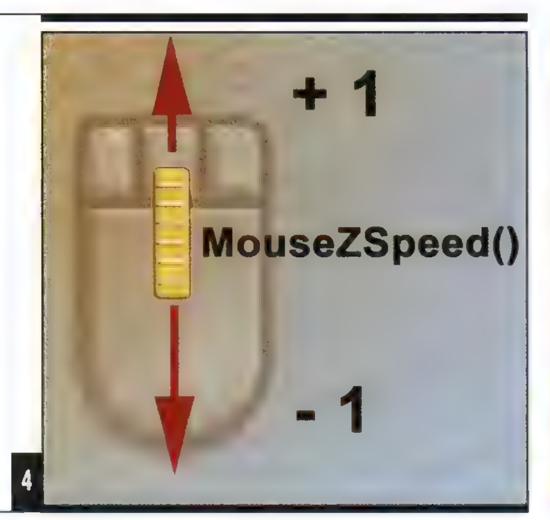
Esquema de los códigos de teclas más usados.

Es importante saber que si no se está usando ningún modo gráfico, el "prompt" aparecerá en la pantalla directamente. Si por el contrario utilizamos modo gráfico, se mostrará en el búfer de dibujo actual.

Otra forma de obtener datos desde el teclado es utilizando "GetKey()". La diferencia con "Input" es notable. Mientras que



Funcionamiento de "MouseX()", "MouseY()" y "MouseZ()".



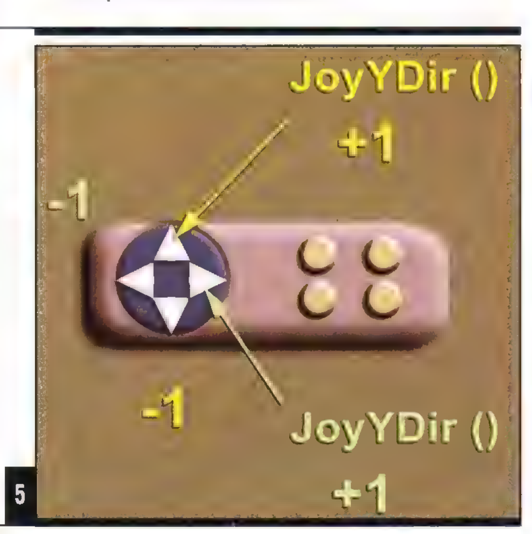
Funcionamiento de "MouseZSpeed()".

```
While Not numero
   Numero=GetKey()
Wend
Print numero
```

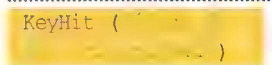
En este pequeño ejemplo, si pulsamos la tecla "A" el programa imprimirá "194" que corresponde a su valor en la tabla ASCII.

Sin embargo, si queremos detectar más de una tecla a la vez o una tecla especial; por ejemplo, "Ctrl + A"o simplemente "Crtl", necesitaremos otro tipo de instrucción. Blitz3D nos soluciona este problema ofreciéndonos los comandos "KeyHit", "KeyDown" y "WaitKey()".

El primero de ellos:



Las instrucciones "JoyXDir()" y "JoyYDir()" son ideales para el manejo de joysticks digitales.



Nos informa de cuántas veces se ha pulsado una determinada tecla. Es ideal, por ejemplo, para detectar la tecla "Escape" para salir de un bucle.

```
While Not KeyHit(1)
    PROCESOS
Wend
End
```

El "Código de la tecla pulsada" es un valor que corresponde a una tecla. En el CD se encuentra la lista de cada uno de estos códigos y la tecla a que corresponde "CODIGOS_TECLADO".

A veces necesitaremos saber qué tecla está siendo pulsada. Para ello utilizaremos el segundo de los comandos que Blitz3D nos proporciona:

```
KeyDown (
        )
```

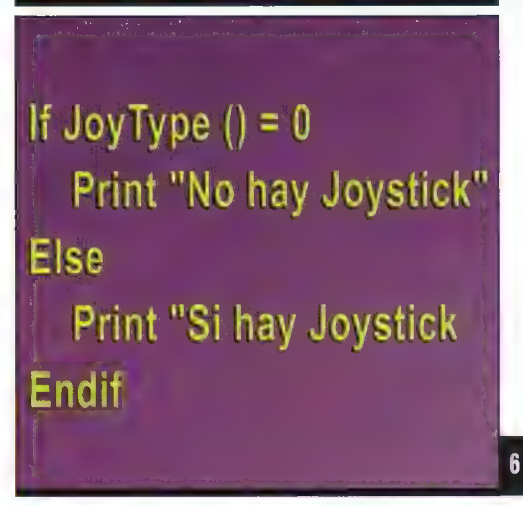
Este comando es el que debemos utilizar si queremos detectar varias teclas a la vez, como por ejemplo, "Ctrl+A" , "Shift + Ctrl + V" o los cursores:

```
...
If KeyDown(29) And
KeyDown(194) Then Print
"Se ha pulsado Ctrl + A"
...
```

Este comando devuelve el valor 0 si no se ha pulsado la tecla y un 1 en caso contrario. Por lo tanto, la anterior sentencia es equivalente a esta otra:

```
If KeyDown(29)=1 And
KeyDown(194)=1 Then Print
"Se ha pulsado Ctrl + A"
```

La última instrucción de que disponemos es "WaitKey()" y se usa



Código para la detectar si hay joystick conectado.

generalmente sola y cuando queremos parar la ejecución del programa hasta que se pulse cualquier tecla. También se puede usar para asignar el código ASCII de una tecla a una variable.

Cada vez que pulsamos las teclas en operaciones de input van creando una cola de espera en la memoria. Para borrar esta porción de memoria se utiliza "FlushKeys".

FUNCIONES DE RATÓN

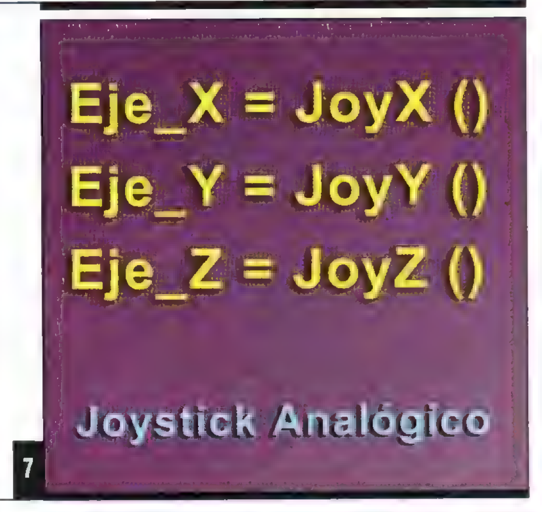
Blitz3D también permite controlar el manejo de un ratón de hasta 3 botones y rueda deslizadora.

Al igual que ocurre con los comandos para detectar la pulsación del teclado "KeyHit" y "KeyDown", también los disponemos para el ratón: "MouseHit" y "MouseDown":

```
MouseHit (                )
```

Este comando cuenta las veces que se ha hecho clic con el ratón desde la última vez que se llama. El parámetro "Nº del botón" puede valer 1, 2 ó 3 dependiendo de qué botón se ha pulsado:

- 1 ➠ Botón izquierdo
- 2 ➠ Botón derecho
- 3 ➠ Botón central



Instrucciones correctas para detectar el movimiento de un Joystick Analógico.

Eje_X = JoyXDir ()
Eje_Y = JoyYDir ()
Eje_Z = JoyZDir ()
Joystick Digital
8

Instrucciones correctas para detectar el movimiento de un Joystick Digital.

```
While Not MouseHit(1)
    ...
Wend
```

En este ejemplo, hasta que no se pulsa el botón izquierdo no se sale del bucle.

Para detectar si se ha pulsado contamos con el comando "MouseDown":

```
MouseDown (Nº del Botón)
If MouseDown(1) = 1  Print "Se
ha pulsado el botón izquierdo"
```

En ocasiones necesitaremos saber qué botón se ha pulsado. Para ello, disponemos de "GetMouse":

```
      = GetMouse()
While Not KeyHit(1)
 Boton=GetMouse()
 Select Boton
 Case 1: Print "Se ha pulsado
 el botón izquierdo"
 Case 2: Print "Se ha pulsado
 el botón derecho"
 Case 3: Print "Se ha pulsado
 el botón central"
 End Select
Wend
```

Si no se pulsa ninguno devolverá el valor 0.

El equivalente a "WaitKey()" lo encontramos en "WaitMouse()". Su función es exactamente la misma. Supongamos que después de realizar operaciones con los botones del ratón hacemos una petición al usuario en el que se pide que se pulse un botón, por ejemplo, para salir con "WaitMouse()". Debemos entonces borrar la cola de pulsaciones anterior. Por lo tanto, utilizaremos "FlushMouse".

Por defecto, en cualquier aplicación, Blitz3D está continuamente detectando el ratón; es decir, se encuentra en todo momento en pantalla. Lo que ocurre es que, en modo de pantalla completa, se mantiene invisible. Para verlo, debemos asignar a sus coordenadas un gráfico, que, por ejemplo, represente un puntero. Para saber entonces la posición exacta del ratón disponemos de las instrucciones "MouseX()","MouseY()".

```
While Not KeyHit(1)
 ...
 DrawImage puntero, MouseX(),
 MouseY()
 ...
Wend
```

"MouseX()" nos devuelve la coordenada X de la posición del ratón y "MouseY()" la coordenada Y.

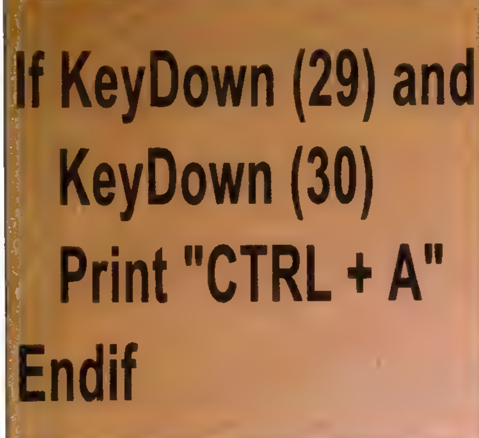
Podemos necesitar, entonces, colocar el puntero en un determinado lugar de la pantalla. Para ello, podemos utilizar "MoveMouse":

```
MouseMove
```

Blitz3D nos permite, además, detectar el deslizamiento de la rueda que algunos modelos de ratón llevan incorporado con la instrucción "MouseZ()". El valor que devuelve "MouseZ()" se incrementará cuando se mueve la rueda hacia delante y decrementará cuando se mueva hacia atrás.

```
While Not KeyHit(1)
 Posicion = MouseZ()
 Print "Posición de la rueda
 ", Posicion
Wend
```

No queda aquí el control que podemos hacer del ratón. También es posible, incluso, detectar los cambios de posición del ratón entre llamadas; es decir, saber dónde estuvo el ratón antes y dónde está ahora. Para obtener esta información tenemos las instrucciones "MouseXSpeed" y



Ejemplo de cómo utilizar "KeyDown" para detectar una combinación de teclas.

```
While Not Salir
  Salir=GetKey()
  Color Rnd(100,255),40,50
  Rect Rnd(10,500),Rnd(10,400),40,50
Wend
```

10

Utilización de "GetKey"para salir de un bucle.

```
While Not KeyHit (1)
  Color Rnd(100,255),40,50
  Rect Rnd(10,500),Rnd(10,400),40,50
Wend
```

11

Otra forma de salir de un bucle al pulsar la tecla "escape". Esta vez con "KeyHit".

"MouseYSpeed". Podemos ver cómo funcionan estos comandos en el ejemplo "ejemplo1.bb".

Cuando se hace una primera llamada a la función, ésta graba la situación del ratón. Al llamarse por segunda vez, devuelve el desplazamiento del ratón entre las dos llamadas.

Con la instrucción "MouseZSpeed()" averiguaremos en qué dirección se ha girado la rueda del ratón.

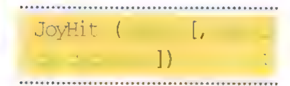
```
= MouseZSpeed()
➡ -1 hacia delante
➡  1 hacia atrás
➡  0 no se ha movido
```

DISPOSITIVOS PARA JUEGOS

El total control de dispositivos para juegos como joystick o gamepad está presente en Blitz3D a través de numerosas funciones. Para empezar, es importante saber si hay algún dispositivo conectado ordenador, y si es así, de qué tipo es. Para averiguar esta información disponemos de la instrucción "JoyType":

```
JoyType ([ ])
```

Esta función devuelve un 0 si no hay nada conectado, un 1 si el joystick es digital y un 2 si es analógico. La opción "Puerto" le indica a Blitz3D que chequee solo en el puerto de conexión especificado. Al igual que ocurre con las funciones del ratón para detectar la pulsación de los botones, disponemos de las mismas funciones para el joystick: "JoyDown", "JoyHit" y "GetJoy":



```
JoyHit ( [,
 ])
```

Devuelve el número de veces que se ha pulsado el botón especificado del joystick.

```
JoyDown ( [,
 ])
```

Detecta cuándo un botón está siendo pulsado. Blitz3D automáticamente puede detectar todos los botones del joystick conectado, ya tenga 3 o 6. Y por último: GetJoy ([Puerto]) devuelve el botón que ha sido pulsado. Para controlar el desplazamiento y dirección de los ejes del joystick ya sea analógico (palanca) o digital (gamepad, botones) disponemos de comandos extremadamente sensibles.

Con "JoyX","JoyY" y "JoyZ".

```
JoyX ([ ])
```

y

```
JoyY ([ ])
```

Devuelven los valores de los ejes X e Y del joystick o gamepad. El valor devuelto es del tipo *float* y va desde –1 hasta 1. La función "JoyZ" es similar a "JoyX" a diferencia de que los valores devueltos representan el máximo (1) o el mínimo (-1) en el desplazamiento.

Por último, podemos saber en qué dirección se han movido los ejes X, Y y Z del joystick mediante los comandos "JoyXDir", "JoyYDir" y "JoyZDir".

Si el joystick se mueve hacia la izquierda "JoyXDir" devolverá el valor entero –1 y si lo hace hacia la derecha, devolverá un 1. Si se mueve hacia arriba "JoyYDir" o JoyZDir" devolverá un –1 y hacia abajo un 1.

Para finalizar, podemos también borrar la cola de botones pulsados que se graba en memoria utilizando la instrucción "FlushJoy".

El uso que le demos a estas funciones dependerá, como es lógico, de nuestras necesidades. Pero hay que tener siempre en cuenta, por ejemplo, en un juego, que la velocidad de respuesta de nuestro programa en las decisiones del usuario es primordial para la jugabilidad. Es preciso, entonces, utilizar los comandos idóneos según las necesidades, así que es importantísimo estudiar todas las posibilidades que Blitz3D nos brinda.

NOTA

Si se quiere detectar joysticks digitales como gamepads es conveniente el uso de las funciones "JoyXDir", "JoyYDir" y "JoyZDir".

En el próximo número...

... entraremos de lleno en el manejo de ficheros.

Grabación y reproducción del movimiento de la cámara

Un elemento muy importante en todo videojuego, y cada vez más utilizado, es la introducción de cinemáticas (animación) en las partidas utilizando el propio motor gráfico del juego.

Estas animaciones pueden ser de muchos tipos dependiendo de la complejidad de las escenas que se quieren reproducir. Desde simples movimientos de cámara hasta complejos trozos de películas mostrando personajes, acción y narrativa. En este tutorial vamos a desarrollar la más sencilla de ellas, el movimiento de la cámara. Aun así, resulta tremendamente útil para un juego o cualquier otra aplicación multimedia, bien para situar la posición de una zona determinada o para realizar un vuelo por una superficie.

SISTEMA DE GRABACIÓN UTILIZANDO MATRICES

La idea básica del proceso se basa en la grabación de la posición y rotación de la cámara al mismo tiempo que es movida por el usuario. El almacenamiento de estos datos se puede hacer de varias formas. Una de ellas podría ser utilizando una matriz para cada dato. Este array se dimensionaría previamente antes del proceso de grabación con una cantidad de posiciones determinada:

```
Duracion=1000 ; Nº Posiciones
Dim PosX#(Duracion),
PosY#(Duracion),
PosZ#(Duracion)
Dim Pitch#(Duracion),
Yaw#(Duracion)
...
```

Bien, este sistema podría funcionar perfectamente. Simplemente, utilizaríamos un contador para desplazar el índice en cada dibujado de la escena y almacenaríamos los datos correspondientes:

```
While Not KeyDown(1)

 MOVER CAMARA

 PosX#(Indice) =EntityX(Camara)
 PosY#(Indice) =EntityY(Camara)
 PosZ#(Indice) =EntityZ(Camara)
 Pitch#(Indice) =EntityPitch(Camara)
 Yaw#(Indice) =EntityYaw(Camara)

 Indice=Indice+1
 If Indice=Duracion FIN GRABACION

 RenderWorld
 Flip
Wend
```

Sin embargo, en nuestro tutorial vamos a utilizar otro procedimiento más elegante, rápido y efectivo utilizando bancos de memoria.

BANCOS DE MEMORIA

El uso de matrices es más lento y consume memoria. Para nuestro proceso de grabación necesitamos un sistema más seguro y que utilice la memoria justa. Para ello, Blitz3D permite escribir y leer directamente en memoria utilizando bancos de memoria.

Para crear un banco de memoria se utiliza el comando "CreateBank()":

```
CreateBank (Tamaño)
```

El tamaño viene determinado por el tipo de datos que se van a almacenar. En la figura 1 se muestra una tabla con la memoria ocupada por cada uno de ellos.

Tipo	Memoria Ocupada	Rango
Byte	1 Byte	0 - 255
Short	2 Bytes	0 - 65535
Int	4 Bytes	-2147483647 2147483647
Float	4 Bytes	$-3{,}4 \times 10^{38}$ $3{,}4 \times 10^{38}$

1

Tabla de la memoria en bytes que ocupa cada tipo de datos y su correspondiente rango de valor.

Para escribir en la memoria disponemos de una instrucción para cada tipo de datos (Ver Fig. 2):

```
PokeByte Banco, Desplazamiento,
Valor
```

Esta instrucción almacena un byte de valor (Valor) en el banco de memoria "Banco". El desplazamiento viene determinado por la cantidad de memoria que necesita; es decir, para un valor "Short" el desplazamiento irá de dos en dos, ya que toma dos bytes para almacenar cada dato. Las siguientes instrucciones son: *PokeShort*, *PokeInt* y *PokeFloat* (Ver Fig. 2).

DATO	INSTRUCCIÓN
Byte	PokeByte
Short	PokeShort
Int	PokeInt
Float	PokeFloat

2

Instrucciones disponibles para escribir en memoria los distintos tipos de datos.

Para leer un valor almacenado en memoria disponemos de: *PeekByte, PeekShort, PokeInt, PeekFloat*. Por ejemplo (Ver. Fig. 3):

```
PeekByte (Banco, Desplazamiento)
```



SISTEMA DE GRABACIÓN UTILIZANDO BANCOS DE MEMORIA

Vamos a aprovechar el ejemplo del mapeado cúbico del tutorial número 12 para probar nuestro sistema de grabación y reproducción contenido en el CD con el nombre "grabación.bb". Le añadiremos estos procedimientos en el bucle principal.

Antes de entrar en el bucle principal vamos a reservar memoria para los datos con "CreateBank" (Ver Código 1).

Necesitaremos almacenar valores de tipo *float* para las coordenadas y ángulos, así que necesitaremos 4 posiciones de memoria para cada dato. Un cálculo aproximado nos lleva a que un segundo de animación ocupa 38 posiciones de memoria, por lo tanto necesitaremos 4 veces más. Seguidamente, creamos un banco para cada dato. Ya dentro del bucle principal, utilizaremos la variable "Offset" para realizar el desplazamiento en cada fotograma. Básicamente, el funcionamiento se reduce a ir grabando en memoria la posición y ángulos actuales de la cámara una vez pulsado el botón izquierdo del ratón (Ver Fig. 4). (Ver Código 2).

Para reproducirlos, solamente tenemos que situarnos al principio de la memoria grabada poniendo a cero la variable "Offset" y leer con "PeekFloat". (Ver Código 3)

Este sistema puede ser mejorado en muchos aspectos como la suavidad de rotación o la grabación en un fichero de los datos almacenados para su posterior uso. Todo depende de nuestras necesidades.



Instrucciones disponibles para leer de memoria los distintos tipos de datos.

Código 1. Reservamos memoria para los datos

```
Tiempo=Input("¿ DURACION DE LA GRABACION EN SEGUNDOS ? ")
Posiciones=Tiempo*38   ; Nº posiciones 1 seg. aprox. 38 posiciones
Memoria=Posiciones*4   ; 4 bytes por posición
BankMemX=CreateBank(Memoria)         ; Memoria para posición X
BankMemY=CreateBank(Memoria)         ; Memoria para posición Y
BankMemZ=CreateBank(Memoria)         ; Memoria para posición Y
BankMemPitch=CreateBank(Memoria)     ; Memoria para Pitch
BankMemYaw=CreateBank(Memoria)       ; Memoria para Yaw
```

Código 2. Vamos grabando en memoria

```
If Offset>Memoria-8    SW_Grab=False
If SW_Grab
    Offset=Offset+4
    PokeFloat BankMemX,Offset,EntityX(Camara)
    PokeFloat BankMemY,Offset,EntityY(Camara)
    PokeFloat BankMemZ,Offset,EntityZ(Camara)
    PokeFloat BankMemPitch,Offset,EntityPitch(Camara)
    PokeFloat BankMemYaw,Offset,EntityYaw(Camara)
EndIf
```

Código 3. Ponemos a cero la variable "Offset"

```
If SW_Play
  X#=PeekFloat(BankMemX,Offset)
  Y#=PeekFloat(BankMemY,Offset)
  Z#=PeekFloat(BankMemZ,Offset)
  RX#=PeekFloat(BankMemPitch,Offset)
  RY#=PeekFloat(BankMemYaw,Offset)
  PositionEntity Camara,X,Y,Z
  RotateEntity Camara,RX#,RY#,0
  Offset=Offset+4
  If Offset>Memoria-8 SW_Play=False
  If PeekFloat(BankMemX,Offset)=.0
     TurnEntity Camara,IPitch,0,0
     RotateEntity Camara,0,IYaw,0
     PositionEntity Camara,IPosX,IPosY,IPosZ
     SW_PLay=False
  EndIf
EndIf
```

NOTA

El sistema utilizado en este tutorial permite grabar hasta 100 horas de movimientos de cámara de una sola vez ocupando solo 9 MB de RAM.

En el próximo número...

... empezaremos una serie dedicada a la realización de un vídeo de presentación para nuestro juego.

Aventuras gráficas

Las aventuras gráficas son un género muy especial. Es como estar dentro de un libro de aventuras y participar en su argumento. Siempre ha sido ideal para el disfrute de toda la familia porque alimenta la imaginación y carece prácticamente de violencia gratuita. En esta entrega recorreremos un poco su historia y descubriremos el avance que ha sufrido en la última década.

COMIENZOS Y EVOLUCIÓN

Para conocer el comienzo de las aventuras gráficas debemos remontarnos al nacimiento de la aventura propiamente dicha. En los comienzos de los juegos para ordenador, el género de aventuras solo existía de forma conversacional, sin gráficos.

AVENTURAS CONVERSACIONALES

La primera aventura para ordenador fue *Collosal Cave Adventure* de Crowther y Woods en 1976. Estaba escrita en FORTRAN y no poseía gráficos algunos. Las diferentes situaciones eran explicadas en forma de texto en la pantalla y el jugador interactuaba por medio de acciones escritas utilizando verbos. Por ejemplo, para desplazarnos se utilizaban: "Subir","Norte","Dentro", etc. y para interactuar con los objetos se usaba el verbo más un sustantivo como: "Coger llave" o "Abrir puerta". Este sistema era muy simple pero eficaz y simulaba los libros interactivos de aventuras. Este tipo de juegos abundaba en los ordenadores de la época. Solo se necesitaba un teclado para jugarlos y conocimientos de inglés.

Posterior a *Adventure* apareció la serie *Zork* (traducida al castellano), *Planetfall* y *Leather Goddesses of Phobos*. Poco a poco empezaron a aparecer los gráficos y los juegos adquirieron un sistema híbrido en el que se seguían utilizando comandos escritos como interfaz de usuario y gráficos para describir las situaciones.

NACE LA AVENTURA GRÁFICA

La persona responsable de crear la primera aventura realmente gráfica fue *Roberta Williams* en el año 1980 cuando publicó para Apple II el juego *Mystery House*. Williams, además, dotaba a sus títulos de un carácter muy personal con historias complejas y absorbentes.

Todavía los juegos de aventuras carecían de movimiento alguno. Pero de nuevo Williams junto a su marido deciden desarrollar esta nueva posibilidad. Fundan la empresa Sierra y publican la serie de aventuras más popular de todos los tiempos, *King´s Quest*, en 1984. Estos títulos seguían siendo híbridos pero los personajes del juego ya poseían movimiento.

Aprovechando las posibilidades gráficas del PC empezaron a aparecer aventuras de tipo híbrido con más movimientos y con pequeñas secuencias de animación como la *Leisure Suite Larry* (1987). También se versionaban para PC títulos de los 8 bits como *The Hobbit*. En España llegaban algunos títulos de la mano de Dinamic como *Yength* (1984) o *Arquímedes XII*. También se desarrollaban títulos totalmente en castellano, algunas veces versiones de juegos anglosajones y otras, pro-



1

Mystery House (Arriba) Primera aventura gráfica. King´s Quest (abajo).



2

Otros éxitos de Sierra On Line: La serie Space Quest (arriba) y Gabriel Knight (abajo).



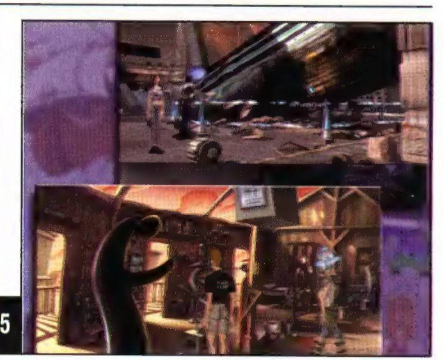
3

La serie de aventuras gráficas más famosa del mundo: Monkey Island.



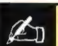
4

El tesoro de la isla Alcachofa (arriba) y Sam & Max (abajo), dos ejemplos de perfecta animación en 2D.



5

Dos ejemplos de aventuras modernas con técnicas en 3D. (Arriba) The Longest Journey y Runaway: a road adventure (abajo).



LA BIOGRAFÍA...

ROBERTA WILLIAMS

Creadora de las **aventuras gráficas**

Empezó en el mundo de las aventuras jugando con *Colosal Cave Adventure* en papel. Le gusta comer bien y ver películas. Pero además de esto le gusta a hacer aventuras gráficas. Junto a su marido Ken fundaron Sierra On-Line después de escribir la primera aventura gráfica de la historia: *Mystery House* en 1980. Sus historias se caracterizan por estar siempre orientadas a todo el público y su mayor logro ha sido la serie *King's Quest* con más de ocho entregas. Sus esfuerzos actuales se centran en las aventuras gráficas en internet.



Roberta Williams

ductos propios. Y todo de la mano de la compañía Aventuras AD, creadores de títulos sólidos como *La aventura original* (1988), *La diosa de Cozumel* (1990) o *Chichén Itzá* (1992)

Sierra seguía siendo la principal productora de aventuras gráficas para ordenador con títulos impecables considerados clásicos del género como la serie *Space Quest o Gabriel Knight*. El primer título de *Space Quest* se basaba todavía en un sistema híbrido pero con la posibilidad de mover el personaje con los cursores del teclado. Las órdenes se seguían dando a través de comandos. Pero cada vez se usaban verbos (cada vez con más posibilidades) para interactuar con los demás elementos del juego.

Otra compañía entró en el mundo de las aventuras gráficas para proporcionar títulos ya clásicos para los PC como *Maniac Mansion*, nos referimos a Lucas Arts. Después de esta aventura, desarrolla el primero de una saga de títulos que marcarían historia: *The Secret of Monkey Island* (1990). Poseía gráficos de 16 colores (aunque apareció posteriormente con 256) con una animación destacable en todos sus aspectos. Era el título que más gráficos utilizaba y sin duda, con el guión más audaz y divertido del género. A partir de los noventa el sistema de interacción a través de comandos verbales se sustituyó por el uso exclusivo del ratón. La riqueza de posibilidades se reducía a acciones determinadas impuestas por los gráficos de la pantalla. Las aventuras gráficas empezaron a coger un rumbo de éxito imparable. Los guiones se hacían más densos y complejos gracias al uso de sistemas de almacenamiento masivo como el CD. Las empresas pioneras optaron por completas bandas sonoras y gráficos espectaculares. Pero todo evoluciona y la tercera dimensión llamaba a las puertas del género cada vez más.

Pero el clásico 2D seguía produciendo títulos con animaciones de calidad como: *El tesoro de la Isla Alcachofa* de Alcachofa Soft o *Sam & Max* y *Day of the Tentacle* (1993, Lucas Arts). Incluso se desarrollaron títulos cuyos protagonistas eran actores reales como la aventura *Morpheus* (2000).

EL PASO A LAS 3D

Definitivamente las mejoras técnicas y cómo no, las exigencias del mercado, llevaron a la clásica representación en 2D de las aventuras gráficas a los entornos en 3D. Sin embargo, este paso no fue del todo inmediato y ni siquiera absoluto en un principio, ya que se opta en muchas ocasiones por el uso híbrido de las dos representaciones mezclando fondos prerenderizados con personajes en 3D y viceversa. Aun así, hoy día prevalecen las 3D e incluso clásicos como *Simon* o *Mokey Island* se desarollan completamente en 3D. A partir del 2000 se desarrollan menos aventuras gráficas pero la mayoría son producciones enormes con guiones extensísimos y complejos y con gráficos muy detallados en 3D en las que incluso se podía rotar la cámara 360 grados como en *Myst III: Exile*. Encontramos títulos como *Schizm: Misterious Journey* (LK, Avalon, 2002) o *The Longest Journey* (Fun Com, 2000).

Pocas compañías se aventuraron a ir mas allá de lo impuesto gráficamente, a excepción de Péndulo que desarrolló *Runaway: A Road Adventure* (Dinamic Multimedia, 2000). Un título en el que se utilizan técnicas 3D pero dotando a los gráficos de un aspecto *cartoon* (dibujos animados) y con grandes resoluciones de pantalla.

En el próximo número...

... entraremos en el mundo de la simulación con el género de simuladores de vuelo.

Cuestionario Videojuegos



Preguntas

1. Escribe un bucle en el que salimos pulsando la tecla "Q".
2. ¿Qué comando debemos utilizar para saber en qué dirección se ha movido la rueda que algunos tipos de ratones llevan como tercer botón?
3. ¿Cómo podemos situar un grupo de objetos aleatoriamente en una zona determinada del terreno?
4. Cuando leemos el fichero que contiene los datos de una zona editada, ¿cómo podemos controlar cuándo no queda ningún dato más en el fichero?
5. ¿Con qué herramientas podemos crear un cubo y situarlo frente a la cámara?
6. ¿Dónde podemos asignar y modificar una textura y cómo se accede?
7. En Anvil Studio, nos situamos en el modo *Piano Roll* de la sección *Compose*. ¿Desde dónde podemos seleccionar un grupo de notas?
8. ¿Cuáles son los procedimientos para grabar una pista del CD en una pista de audio de Anvil Studio?
9. ¿Qué dos sistemas podemos utilizar para grabar y leer datos?
10. ¿Cómo podemos escribir y leer directamente en memoria?

Respuestas al cuestionario 14

▷ **1.**
```
AmbientLight 100,20,50
Direccional=CreateLight()
Puntual=CreateLight(2)
Foco=CreateLight(3)
```

▷ **2.** Mediante la instrucción
```
LightConeAngles Luz, Angulo interior#, Angulo exterior#
```

▷ **3.** Es una zona del entorno 3D donde se crean, evolucionan y destruyen partículas generalmente formadas por sprites.

▷ **4.**
```
Type emisor
Field X_Emisor#,Y_Emisor#,Z_Emisor#     ; Posición del emisor
Field RangoX#,RangoY#,RangoZ#           ; Tamaño del emisor
Field Entidad                           ; Entidad Sprite que forma la partícula
Field Vida                              ; Vida de la partícula una vez emitida
Field Angulo_Rotacion#                  ; Angulo de rotación de cada partícula
Field Xvel#,Yvel#,Zvel#                 ;  Velocidad inicial de las partículas
End Type
```

▷ **5.** Se puede obtener con el programa LithUnwrap seleccionando todo el modelo y eligiendo la opción *Tools / UVMapping*.

▷ **6.** Aplicándola a una primitiva como un cubo, acercando una de sus caras a la cámara y guardando la imagen del render resultante.

▷ **7.** Nos situamos en *Compose* en el modo *Staff*. Nos situamos al comienzo del compás a partir del cual queremos insertar los compases. Pulsamos dos veces sobre el botón de insertar y escribimos "5" en la casilla de petición.

▷ **8.** Dejando pulsada la tecla "Ctrl" y sin dejar de hacer clic sobre la nota, desplazar el ratón.

▷ **9.**
```
Function ordenar_tabla()
  For n= 0 To  19
    For m= 0 To 19
      If Numero(m) < Numero(n)
        Aux_Numero= Numero (m)
        Numero (m)= Numero (n)
        Numero (n)=Aux_ Numero
      EndIf
    Next
  Next
End Function
```

▷ **10.**
```
Function Insertar_en_Tabla(numero)
  Aux_Numero=Numero
  For n=19 To 0 Step -1
    If Aux_Numero>Numero(n)
      Numero(n)=Aux_Numero
      Return
    EndIf
  Next
End Function
```

Contenido CD-ROM 15

▶ AUDIO

■ Tracktion 1.1.0



Controla proyectos musicales de principio a fin valiéndote de esta aplicación.

■ 1st Sound Recorder

Puede grabar en formato WAV o MP3 cualquier sonido que se produzca en tu ordenador, de forma automática y con alta calidad.

■ Advanced Sound Recorder 3.1

Esta aplicación complementa la anterior; es muy sencilla de usar y te permitirá capturar sonido fácil y rápidamente.

■ All Recorder 1.6.1

Graba, edita y modifica cualquier sonido capturado desde micrófono u otros programas en múltiples formatos.

■ Music MasterWorks 3.75

Potente programa para edición y secuenciamiento de archivos MIDI.

■ SpinStation Jukebox

Conviértete en un auténtico DJ usando todas las opciones de esta completa herramienta.

▶ DISEÑO 2D

■ Photomatte Lab 1.0



Completo programa para manipular imágenes y cambiarles las texturas.

■ Art Directors Toolkit 3

Aplicación directa cargada de prestaciones que te permite hacer un montón de cosas con tus imágenes.

■ DigiPicNamer 1.0

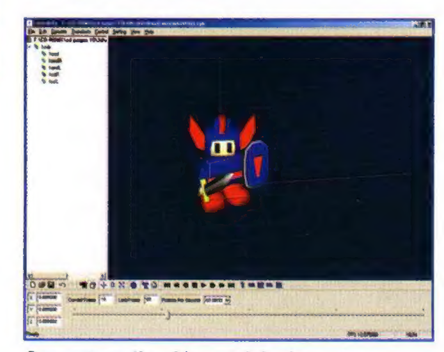
Si vas a manejar muchas imágenes al mismo tiempo, y quieres renombrarlas, podrás hacerlo de golpe con esta utilidad.

■ Photopleasure

Crea álbumes de imágenes o impórtalos a través de este simpático programa.

▶ DISEÑO 3D

■ Cyberdelia 2.5



Con esta aplicación modelarás en tres dimensiones de un modo muy sencillo.

■ 360 Professional Suite

Crea completas panorámicas en tres dimensiones fácil y rápidamente.

■ Bryce 5

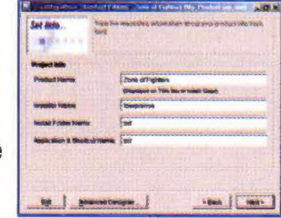
Nueva oportunidad de conseguir este programa, que usamos activamente en el curso.

■ Extreme Morph 3D 1.1

Otro programa de modelado 3D, para realizar primitivas y texturizarlas.

▶ PROGRAMACIÓN

■ InstallAnywhere 5



Simplifica el proceso de instalar en cualquier plataforma, y se hace cargo de todos los detalles.

■ Astrum InstallWizard

Con esta aplicación podrás crear el instalador de tu juego en pocos minutos.

■ Fresh Breeze Intelligent Editor

Editor inteligente con el que podrás usar un montón de lenguajes distintos de programación.

■ PatchWise 3.25

Crea completos proyectos de principio a fin sin ninguna dificultad.

■ POV-Ray 3.5

Uno de los trazadores de rayos más célebres, que nos permite representar escenas imaginarias definidas mediante el uso de unos modelos.

▶ JUEGOS

■ Runaway



Demo del estupendo juego en tres dimensiones con aspecto de dibujos animados.

■ King's Quest 1

Demo del clásico juego de Roberta Williams con el que tan buenos momentos hemos pasado.

■ Leisure Suite Larry

Juego muy divertido que narra los escarceos eróticos de un simpático personaje.

■ Maniac Mansion

Otro de los clásicos dentro de las aventuras gráficas que todos recordamos.

■ Day of the tentacle

Aventura gráfica muy adictiva y divertida.

■ Zone of Fighters

Como siempre, nuestro juego tal y como va quedando.

▶ VÍDEO

■ FlyVCD 2.5.0

Crea vídeos fácilmente a partir de imágenes estáticas.

■ LEAD MCMP_MJPEG Codec

Conversor de formatos de vídeo muy sencillo de usar.

■ MovieWow



Con esta divertida aplicación podrás crear spots comerciales de tu juego.

▶ EXTRAS

En este apartado encontrarás todos los ejemplos de los que hablamos en el coleccionable, para que no pierdas detalle.